

Classification

Types & Algorithms

Classification?

Classification (การจำแนกประเภทข้อมูล)

ปัญหาในการจำแนกประเภทคือเมื่อตัวแปรเวลาต์พุตเป็นหมวดหมู่ เช่น “เสือ” และ “ช้าง” หรือ “โรค” และ “ไม่มีโรค”



Tiger



Elephant

Classification Example.

เราสอนเด็ก ด้วยการชี้ภาพให้ที่ไม่เด็กไม่เคยเห็นดู ว่าตัวไหนแมว ตัวไหนไม่ใช่แมว ซึ่งไป 2-3 วัน ให้เด็กได้เจอสัตว์หลายประเภท วันที่ 4-5 เอาแมวที่เด็กไม่เคยเห็นมาให้ดู รวมกับสัตว์อื่นๆ แล้วให้เด็กตอบว่าตัวไหนแมว

ถ้าถูกแปลว่าการสอนของเรามีประสิทธิภาพ



cat



Not cat



Not cat



Not cat



Not cat



cat



Not cat

ผลลัพธ์ที่ได้จากการสอนเด็กแบบ Classification ที่ไม่ซับซ้อน

Classification Example.

หากตอนเรารสอน นอกไปเลยกว่าเป็น แมว ไก่ หมู ลิง เด็กก็จะตอบได้มากกว่าแค่แมว หรือไม่ใช่แมว ซึ่งวิธีนี้อาจจะต้องใช้กระบวนการสอนที่มีความชับช้อนมากขึ้นไปอีก



cat



chicken



pig



frog



monkey



cat



Penguin

ผลลัพธ์ที่ได้จากการสอนเด็กแบบ Classification ที่ชับช้อน

Classification Types

01

Binary
classification

การจำแนกแบบ二元分類

02

Multi-Class
Classification

การจำแนกประเภทหลายคลาส



01 Binary classification

การจำแนกแบบไบนารี

Let's Get
Started

Binary classification (การจำแนกแบบไบนารี)

เปรียบเทียบให้เข้าใจง่ายที่สุดก็คือ ตัวแปรที่แบ่งเป็นเพียงสองหมวดหมู่ เช่น ผลลัพธ์แบบ “ใช่” หรือ “ไม่ใช่” ซึ่ง หรือ “ไม่ซึ่ง” หากเปรียบเป็นตัวเลขก็คือ 0 กับ 1 อัลกอริทึมที่ใช้คู่กับการจำแนกแบบไบนารีก็มี k-Nearest Neighbors ,Decision Trees (ต้นไม้ตัดสินใจ) Naive Bayes หรือ SVM



Elephant?

Yes
or
No

02

Multi-Class Classification

การจำแนกประเภทหลายคลาส

Let's Get
Started

Multi-Class Classification (การจำแนกประเภทหลายคลาส)

มีหมวดหมู่มากกว่าสอง ยกตัวอย่างเช่น รูปภาพรูปหนึ่งสามารถมีรูปดอกไม้ ห้องฟ้า ก้อนเมฆได้ แต่รูปภาพนั้นจะจัดว่าเป็นหมวดหมู่รูปวด รูปถ่าย หรือรูปเสีย Multi-Class Classification จะจำแนกว่ารูปนั้นเป็นรูปวด รูปถ่ายหรือรูปเสียโดยเดลีที่ใช้คู่กับการจำแนกประเภทหลายคลาสก็สามารถใช้อัลกอริทึมคล้ายกันที่ใช้กับการจำแนกแบบใบหนารีได้

โดยอาศัยกลยุทธ์การแปลงคลาสแบบ

- One-vs-All(OvA)
- One-vs-One(OvO)



One-vs-All(OvA)

One-vs-All (OvA) คือ การแบ่งปัญหาออกเป็น 1 ต่อ อื่นๆ ยกตัวอย่างเช่น หากเรามีหมวดหมู่สัตว์อยู่ 3 หมวดคือ ช้าง ไก่ และหมู เราจะแบ่งปัญหาการจำแนกแบบใบหน้าเป็น 3 ปัญหาคือ 1 จำแนกช้างออกจากทั้งหมด 2 จำแนกไก่และ 3 จำแนกหมู



Elephant?



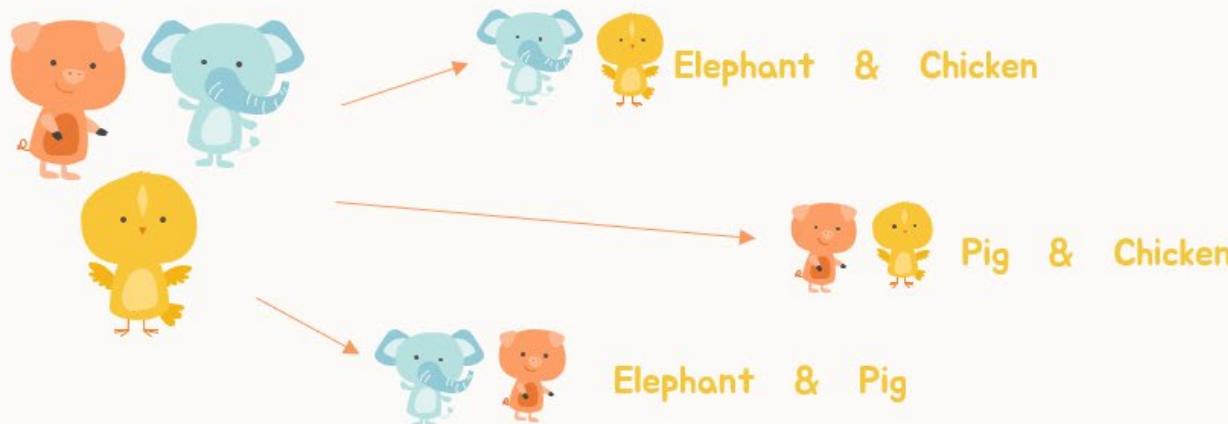
chicken?



Pig?

One-vs-One(OvO)

One-vs-One(OvO) คือ การแบ่งปัญหาออกเป็นคู่ ๆ ยกตัวอย่างเช่นหมวดหมู่สัตว์อยู่ 3 หมวดคือ ช้าง ไก่ และหมู ก็คือแบ่งปัญหาเป็น 1 จำแนกระหว่าง ช้างกับไก่ 2 จำแนกระหว่าง ช้างกับหมู 3 จำแนกระหว่าง ไก่กับหมู



Classification Algorithm

01 Decision tree

04 Naïve Bayes

02 Support Vector Machine

05 XGBoost

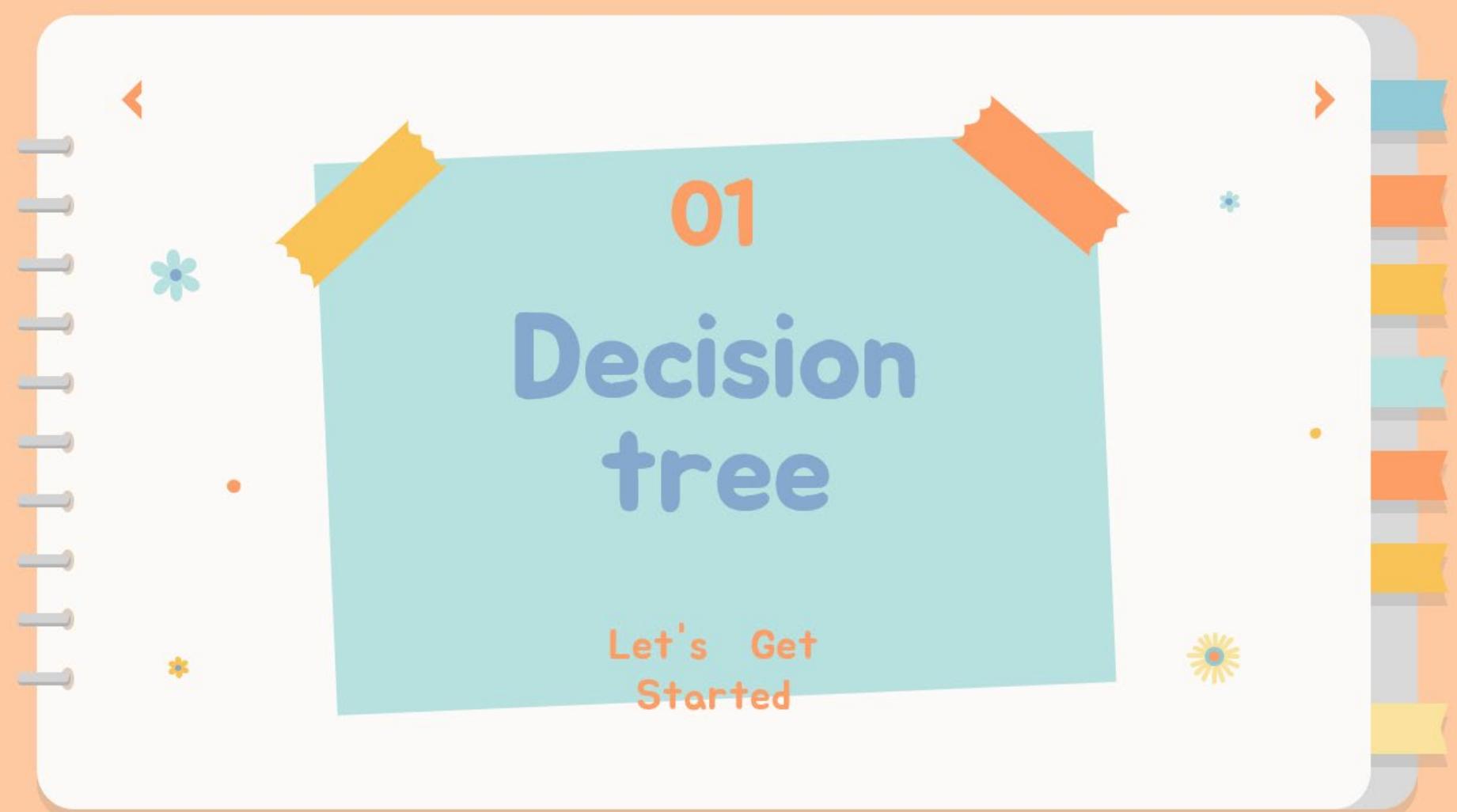
03 K-Nearest Neighbors

06 Random forest

01

Decision tree

Let's Get
Started



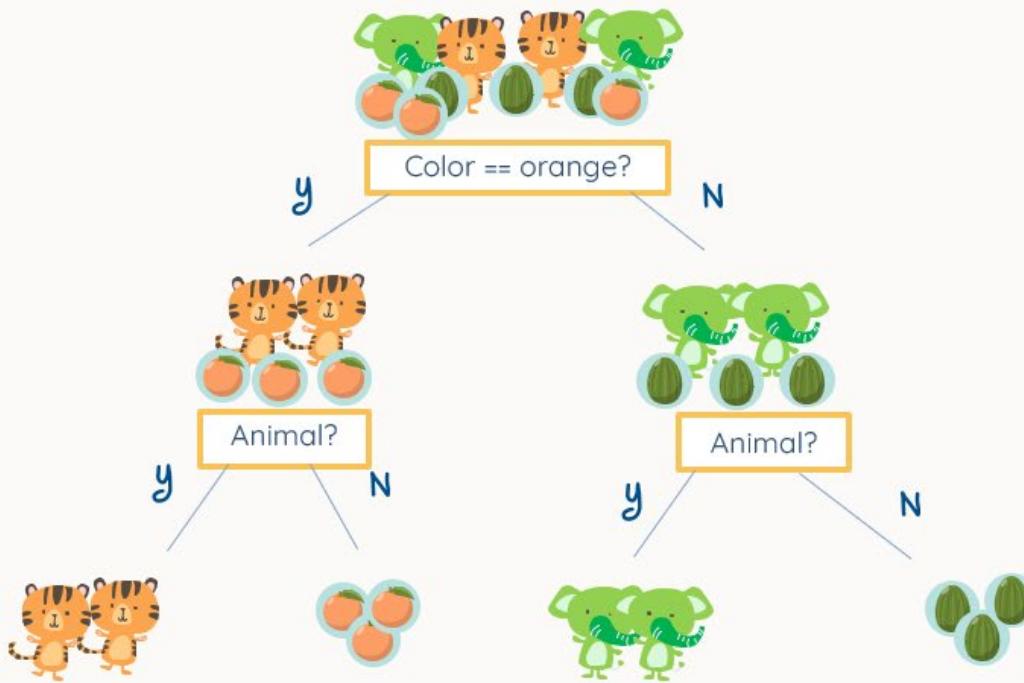
Decision tree

Decision tree คือ model แบบ rule-based คือ สร้างกฎ if-else จากค่าของแต่ละ feature โดยไม่มีสมการมาคำนวณความสัมพันธ์ระหว่าง feature & target สิ่งที่สำคัญในการสร้าง Decision Tree คือ การเลือก split ค่า feature แต่ละครั้ง จะต้อง minimise ค่าของ cost function ให้น้อยที่สุด (regression – mse, classification- impurity, entropy)

Decision tree

Decision tree สามารถอธิบายได้ด้วยรูปแบบของ “TREE” นั้นคือมี Node พ่อเป็นตั้ง
คำถามว่าใช่หรือไม่ Node ลูกตัวแรกอาจจะเป็นใช่ อีกตัวจะเป็นไม่ โดยปัจจัยสำคัญในการสร้าง
ไมเดลนี้คือ “ความลึกของต้นไม้” ยิ่งต้นไม้มีลึก (มีจำนวนชั้นที่มาก) ก็จะสามารถได้ลักษณะมากยิ่งขึ้น
แต่ก็จะยิ่ง Overfit มากขึ้น แต่ถ้าจำนวนชั้นที่น้อยไป ก็จะไม่แม่นยำพอจะใช้งาน

Decision tree



Decision tree

```
In [1]: import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import sklearn.datasets as datasets  
%pylab inline
```

Populating the interactive namespace from numpy and matplotlib

```
In [2]: iris = datasets.load_iris()  
XX = pd.DataFrame(iris.data, columns=iris.feature_names)  
yy = pd.DataFrame(iris.target)
```

```
In [3]: from sklearn.model_selection import train_test_split  
  
X_train, X_test, y_train, y_test = train_test_split(XX, yy, test_size=0.3, random_state=0)  
  
print('There are {} samples in the training set and {} samples in the test set'.format(X_train.shape[0], X_test.shape[0]))
```

There are 105 samples in the training set and 45 samples in the test set

```
In [4]: from sklearn.preprocessing import StandardScaler  
from sklearn.model_selection import cross_val_score  
  
sc = StandardScaler()  
sc.fit(X_train)  
X_train_std = sc.transform(X_train)  
X_test_std = sc.transform(X_test)
```

Import

- Numpy
- pandas
- Sklearn datasets

Load data

ตอนนี้ไปเริ่มสร้างโมเดลต้นไม้แล้ว โดยจะใช้ `StandardScaler` ในการนำค่ามาปรับเปลี่ยนให้เป็นค่าที่อยู่ในช่วง [-1, 1] ให้กับตัวแปร `X_train` และ `X_test` ก่อน ซึ่ง `StandardScaler` คือโมเดลที่จะคำนวณค่าเฉลี่ยและส่วนเบี่ยงเบนมาตรฐานของข้อมูลที่ได้รับ และใช้ค่าเฉลี่ยและส่วนเบี่ยงเบนมาตรฐานนี้ในการคำนวณค่า `X_train_std` และ `X_test_std` ที่คำนวณ出来

Decision tree

```
In [5]: #Applying Decision Tree
from sklearn import tree

#Create tree object
decision_tree = tree.DecisionTreeClassifier(criterion='gini')

#Train DT based on scaled training set
decision_tree.fit(X_train_std, y_train)

#print performance
print('The accuracy of the Decision Tree classifier on training data is {:.2f}'.format(decision_tree.score(X_train_std, y_train)))
print('The accuracy of the Decision Tree classifier on test data is {:.2f}'.format(decision_tree.score(X_test_std, y_test)))
```

The accuracy of the Decision Tree classifier on training data is 1.00
The accuracy of the Decision Tree classifier on test data is 0.98

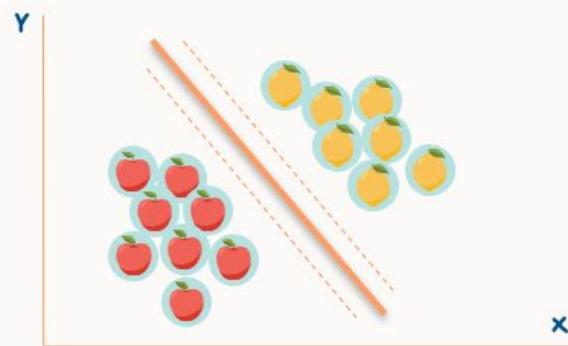
02

Support Vector Machine

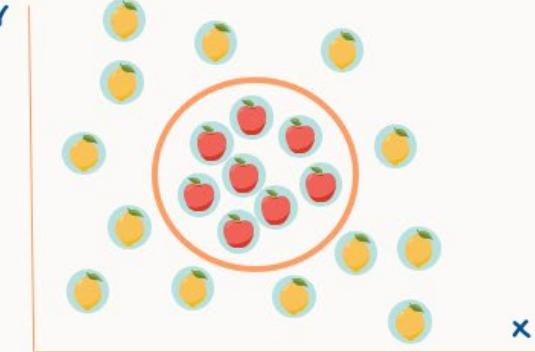
Let's Get
Started

Support Vector Machine

Support Vector Machine จะเป็นการจัดกลุ่มข้อมูล Classification โดยการแบ่ง Class ของข้อมูลออกจากกัน ซึ่งสามารถใช้การแบ่งด้วยสมการเชิงเส้นได้ทั้ง Linear และ Non Linear



Linear separable

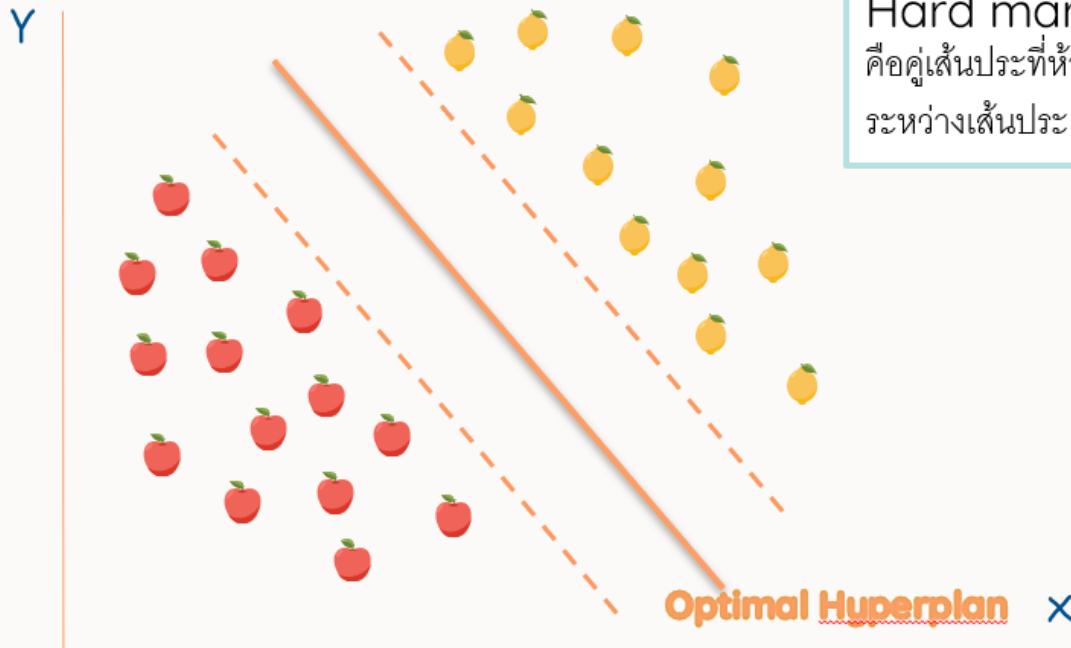


Non linear separable

Support Vector Machine

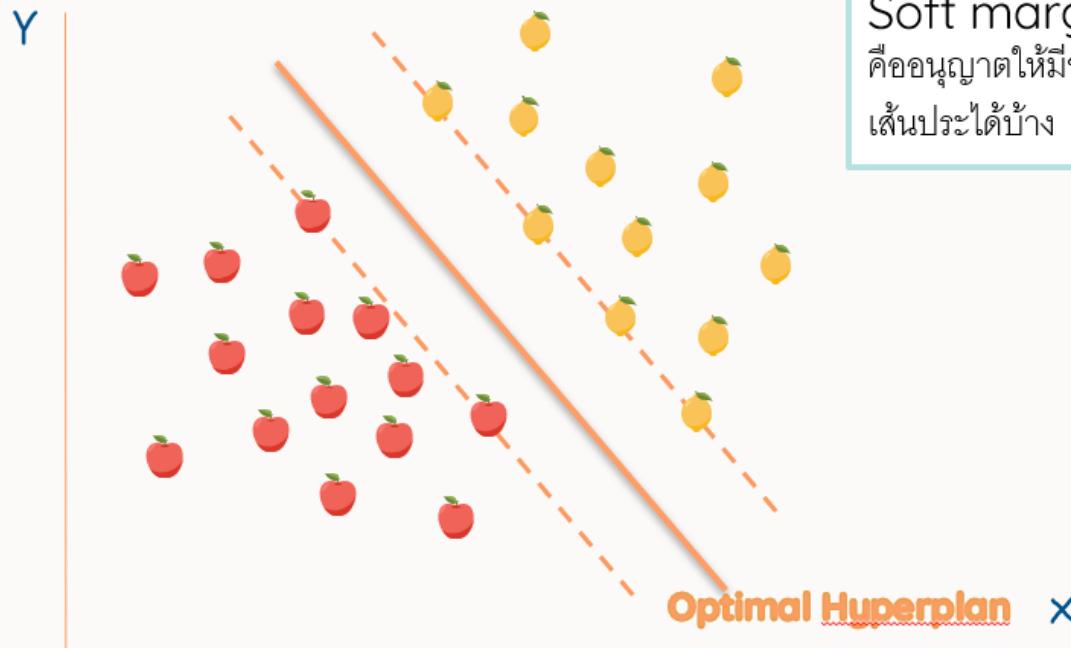
Support Vector Machine การจัดกลุ่มข้อมูล Classification 'ไม่ใช้เรื่องยาก สามารถแบ่งข้อมูลได้หลาย Model โดยใช้สมการเส้นตรง Linear แต่ปัญหาคือแล้ว Model ไหนที่ดีที่สุด ซึ่ง Support Vector Machine จะใช้ในการหา Hyperplan ซึ่งคือ Model ที่ดีที่สุด จากระยะห่างสูงสุดของแต่ละ Class ข้อมูล Maximum Margin ในแต่ละมิติ N - Dimension โดย N เป็นจำนวนของ Feature

Support Vector Machine



Hard margin classification
คือคู่เส้นประที่ห้ามไม่ให้มีจุดข้อมูลอยู่ในพื้นที่ระหว่างเส้นประ

Support Vector Machine



Soft margin classification
คืออนุญาตให้มีข้อมูลอยู่ในพื้นที่ระหว่าง
เส้นประได้บ้าง

Support Vector Machine

```
In [1]: import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import sklearn.datasets as datasets  
%pylab inline
```

Populating the interactive namespace from numpy and matplotlib

```
In [2]: iris = datasets.load_iris()  
XX = pd.DataFrame(iris.data, columns=iris.feature_names)  
yy = pd.DataFrame(iris.target)
```

```
In [3]: from sklearn.model_selection import train_test_split  
  
X_train, X_test, y_train, y_test = train_test_split(XX, yy, test_size=0.3, random_state=0)  
  
print('There are {} samples in the training set and {} samples in the test set'.format(X_train.shape[0], X_test.shape[0]))
```

There are 105 samples in the training set and 45 samples in the test set

```
In [4]: from sklearn.preprocessing import StandardScaler  
from sklearn.model_selection import cross_val_score  
  
sc = StandardScaler()  
sc.fit(X_train)  
X_train_std = sc.transform(X_train)  
X_test_std = sc.transform(X_test)
```

Import

- Numpy
- pandas
- Sklearn datasets

Load data

ต่อไปคือสิ่งอ่อนเจ็กชื่นจากคลาส StandardScaler จากนั้นก็มาออบเจกต์ตัวนี้ไปที่โดยเริ่มจากใช้เมธอด fit เพื่อนำข้อมูลมาหาค่าเฉลี่ยและส่วนเบี่ยงเบนมาตรฐานเพื่อนำมาใช้ในการแปลงต่อไปจากนั้นจึงใช้เมธอด transform เพื่อทำการแปลงค่าให้เป็นค่าที่ปรับเป็นมาตรฐานแล้ว ผลที่ได้คือได้ X_train_std และ X_test_std ที่คำนวณเองได้

Support Vector Machine

```
In [5]: #Applying SVC (Support Vector Classification)
from sklearn.svm import SVC

svm = SVC(kernel='rbf', random_state=0, gamma=.10, C=1.0)
svm.fit(X_train_std, y_train)

print('The accuracy of the SVM classifier on training data is {:.2f}'.format(svm.score(X_train_std, y_train)))
print('The accuracy of the SVM classifier on test data is {:.2f}'.format(svm.score(X_test_std, y_test)))
```

The accuracy of the SVM classifier on training data is 0.97
The accuracy of the SVM classifier on test data is 0.98

03

K-Nearest Neighbors

Let's Get
Started

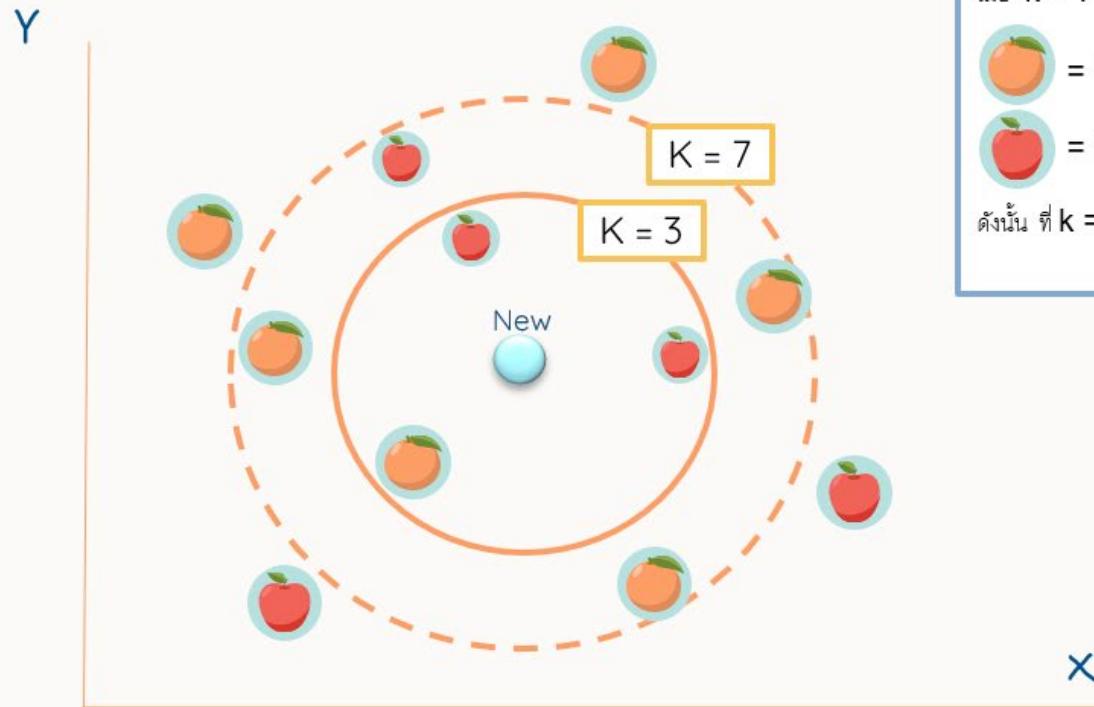


K-Nearest Neighbors (K-NN) .



K-Nearest Neighbors (K-NN) เป็นวิธีในการจำแนกประเภทของข้อมูล วิธีการนี้ที่เป็นที่นิยมในการจำแนกประเภทข้อมูล เพราะลักษณะพิเศษของ K-NN คือ Lazy-Learning ซึ่งเหตุที่ถูกเรียกว่าเป็น Lazy-Learning นั้นมาจากการรูปแบบในการจำแนกประเภทข้อมูลของ K-NN โดยไม่มีการสร้างโมเดล สำหรับจำแนกประเภทข้อมูลเตรียมไว้ล่วงหน้าเมื่อมีข้อมูลใหม่ที่ต้องการจำแนกประเภท เพียงนำมาเทียบกับข้อมูลเดิมและดูถึงความคล้ายคลึงกันของข้อมูลใหม่และข้อมูลเดิมที่มีความสามารถจำแนกประเภทของข้อมูลใหม่ได้โดยให้เป็นประเภทเดียวกับข้อมูลเดิมที่อยู่ใกล้เคียง

K-Nearest Neighbors (K-NN)



เมื่อ $k = 7$

○ = 4

● = 3

ดังนั้น ที่ $k = 7$ เป็นคลาส



K-Nearest Neighbors

```
In [1]: import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import sklearn.datasets as datasets  
%pylab inline
```

Populating the interactive namespace from numpy and matplotlib

```
In [2]: iris = datasets.load_iris()  
XX = pd.DataFrame(iris.data, columns=iris.feature_names)  
yy = pd.DataFrame(iris.target)
```

```
In [3]: from sklearn.model_selection import train_test_split  
  
X_train, X_test, y_train, y_test = train_test_split(XX, yy, test_size=0.3, random_state=0)  
  
print('There are {} samples in the training set and {} samples in the test set'.format(X_train.shape[0], X_test.shape[0]))
```

There are 105 samples in the training set and 45 samples in the test set

```
In [4]: from sklearn.preprocessing import StandardScaler  
from sklearn.model_selection import cross_val_score  
  
sc = StandardScaler()  
sc.fit(X_train)  
X_train_std = sc.transform(X_train)  
X_test_std = sc.transform(X_test)
```

Import

- Numpy
- pandas
- Sklearn datasets

Load data

ต่อไปคือสิ่งของเรียกว่า scaler จากรากคลาส StandardScaler จากนั้นก็มาออบเจกต์ sc นั้นไปใช้โดยรีเม็ท fit เพื่อนำชั้้อมูลมาหาค่าเฉลี่ยและส่วนเบี่ยงเบนมาตรฐานเพื่อนำมาใช้ในการแปลงต่อไป จากนั้นจึงใช้มอง transform เพื่อทำการแปลงค่าที่เป็นค่าที่ปรับเปลี่ยนมาตรฐานแล้ว ผลที่ได้คือได้ X_train_std และ X_test_std ที่คำนวณเองได้

K-Nearest Neighbors

```
In [5]: #Applying Knn
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors = 7, p = 2, metric='minkowski')
knn.fit(X_train_std, y_train)

print('The accuracy of the Knn classifier on training data is {:.2f}'.format(knn.score(X_train_std, y_train)))
print('The accuracy of the Knn classifier on test data is {:.2f}'.format(knn.score(X_test_std, y_test)))
```

The accuracy of the Knn classifier on training data is 0.97
The accuracy of the Knn classifier on test data is 0.98



04

Naïve Bayes

Let's Get
Started

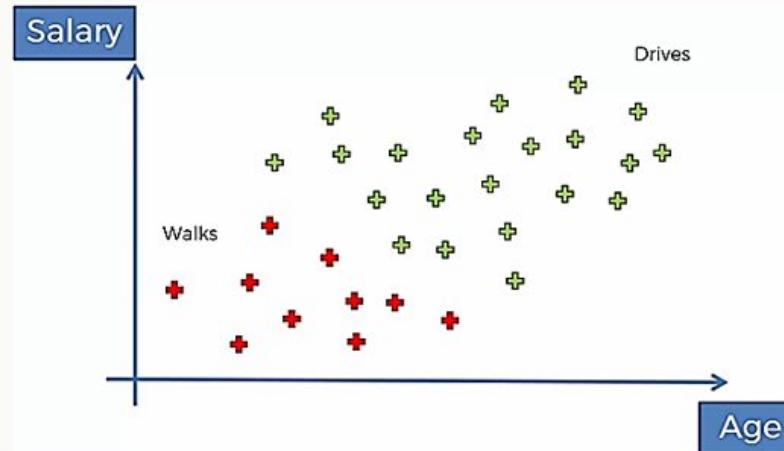
Naïve Bayes

Naive Bayes classification เป็นเทคนิคการจำแนกตามสมมติฐานของความเป็นอิสระระหว่างตัวทำนายซึ่งเรียกว่าทฤษฎีเบย์พูดง่ายๆคือลักษณะ Naive Bayes จะถือว่าการมีอยู่ของคุณสมบัติเฉพาะในคลาสนั้นไม่เกี่ยวข้องกับการมีอยู่ของคุณสมบัติอื่นใด

Naive Bayes นั้น 'ไร้เดียงสา' มากเพระมันตั้งสมมติฐานที่แทบจะมองไม่เห็นในชีวิตจริงและถือว่าคุณสมบัติทั้งหมดเป็นอิสระ

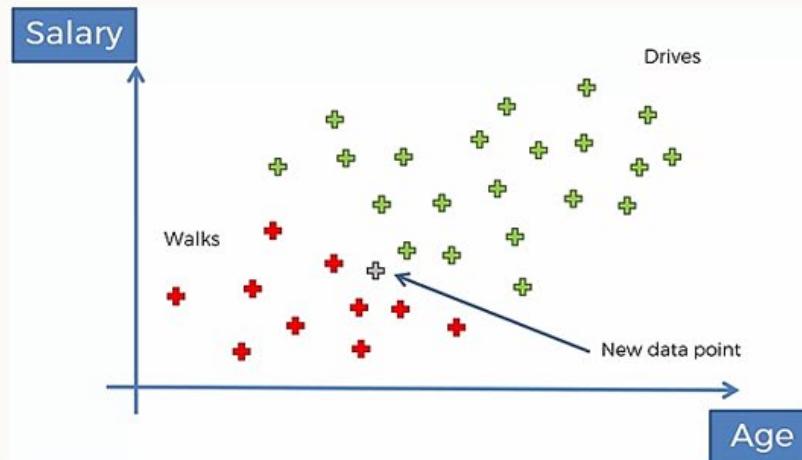
Naïve Bayes

มีชุดข้อมูลชุดหนึ่ง มีสองคอลัมน์และจุดสัมภาระหรือจุดข้อมูลแบ่งออกเป็นสองประเภทที่แตกต่างกันด้วยสีแดงและสีเขียว ดังนั้นแกน X แทนอายุในขณะที่แกน Y แทนเงินเดือน หมวดหมู่ของคนที่เดินไปทำงานของพากษา เป็นสีแดงและสีเขียวคือหมวดหมู่ของคนที่ขับเคลื่อนไปยังที่ทำงานของพากษา



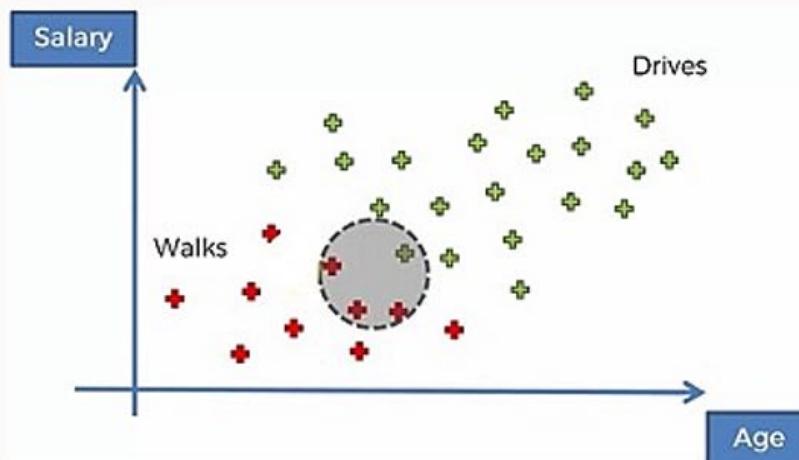
Naïve Bayes

ตอนนี้จะเกิดอะไรขึ้นถ้าเราทำการซื้อข้อมูลใหม่เข้าไปในชุดข้อมูล เราจะจำแนกจุดข้อมูลใหม่นี้อย่างไร เรากำลังจะจัดประเภทจุดข้อมูลใหม่โดยใช้ทฤษฎีของ Naive Bayes เพื่อวินิจฉัยว่ามันอยู่ในประเภทจุดสีแดง หรือสีเขียวันนี้คือบุคคลใหม่นั้นเดินหรือขับรถไปทำงาน



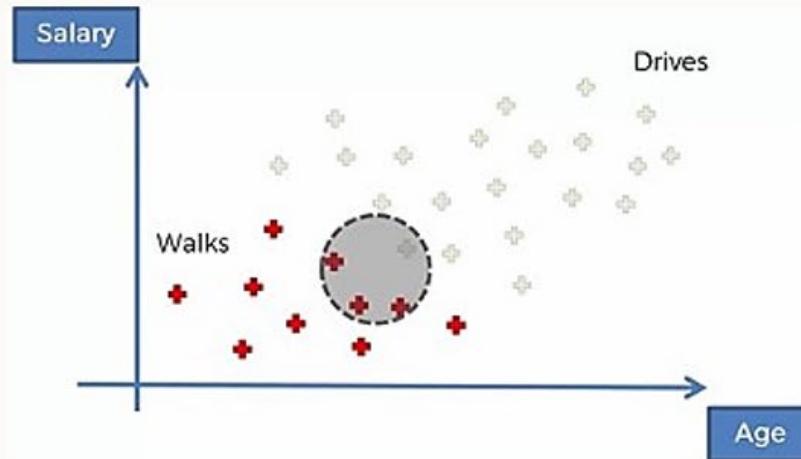
Naïve Bayes

ในขั้นตอนต่อไปเราจะคำนวณความเป็นไปได้ สิ่งแรกที่เราจะทำคือเลือกรัศมีและเราจะดูวงกลมรอบจุดสังเกตหรือจุดข้อมูลใหม่จากนั้นเราจะดูจุดที่อยู่ในวงกลมไม่ว่าจะเป็นข้อมูลใดก็ตาม ในบริเวณใกล้เคียงนั้นจะถือว่าคล้ายกับจุดข้อมูลใหม่ที่เราเพิ่งลงในชุดข้อมูลของเรา



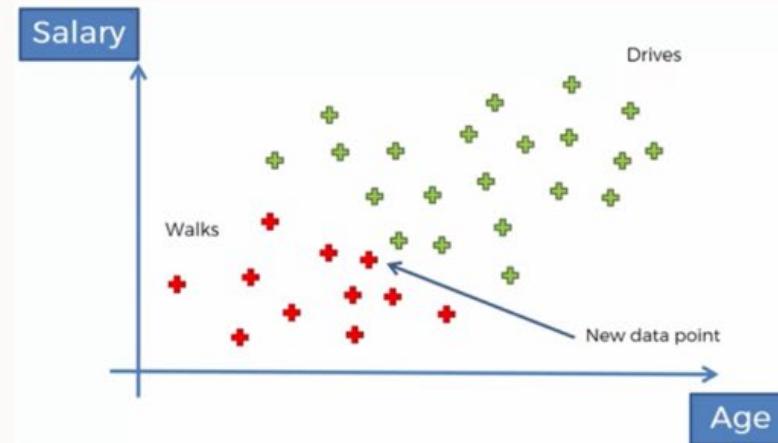
Naïve Bayes

ความเป็นไปได้คือจุดข้อมูลที่สุ่มเลือกจากชุดข้อมูลจากจุดสีแดงคือข้อมูลที่แสดงคุณลักษณะที่คล้ายกับจุดที่เรากำลังเพิ่มลงในชุดข้อมูลของเราหรือกล่าวอีกนัยหนึ่งคือจุดสีแดงที่เลือกแบบสุ่มจะตกลอยู่ในพื้นที่วงกลม



Naïve Bayes

ตอนนี้เราได้สมมติความน่าจะเป็นได้ 0.75 เช่น 75% คือความน่าจะเป็นที่จุดข้อมูลใหม่จะถูกจัดประเภทเป็นบุคคลที่เดินไปทำงาน และ 25% ($1 - 0.75$) คือความน่าจะเป็นที่จุดข้อมูลใหม่จะถูกจัดประเภทเป็นบุคคลที่ขับเคลื่อนให้ทำงาน



ในที่สุดเราจะจำแนกจุดข้อมูลใหม่ซึ่งเป็นบุคคลที่เดินไปทำงานเป็นจุดแดง

Naïve Bayes

```
In [1]: import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import sklearn.datasets as datasets  
%pylab inline
```

Populating the interactive namespace from numpy and matplotlib

```
In [2]: iris = datasets.load_iris()  
XX = pd.DataFrame(iris.data, columns=iris.feature_names)  
yy = pd.DataFrame(iris.target)
```

```
In [3]: from sklearn.model_selection import train_test_split  
  
X_train, X_test, y_train, y_test = train_test_split(XX, yy, test_size=0.3, random_state=0)  
  
print('There are {} samples in the training set and {} samples in the test set'.format(X_train.shape[0], X_test.shape[0]))
```

There are 105 samples in the training set and 45 samples in the test set

```
In [4]: from sklearn.preprocessing import StandardScaler  
from sklearn.model_selection import cross_val_score  
  
sc = StandardScaler()  
sc.fit(X_train)  
X_train_std = sc.transform(X_train)  
X_test_std = sc.transform(X_test)
```

Import

- Numpy
- pandas
- Sklearn datasets

Load data

ต่อไปเดี๋ยวสร้างอุปกรณ์ชื่อจากคลาส StandardScaler จากนั้นก็นำอุปกรณ์ตัวนั้นไปใช้โดยเริ่มจากให้มีเมธอด fit เพื่อนำข้อมูลมาหาค่าเฉลี่ยและส่วนเบี่ยงเบนมาตรฐานเพื่อนำมาใช้ในการแปลงต่อไป หากนั้นจึงให้เมธอด transform เพื่อทำการแปลงค่าให้เป็นค่าที่ปรับเป็นมาตรฐานแล้ว ผลที่ได้คือได้ X_train_std และ X_test_std ที่คำนวณเองได้

Naïve Bayes



```
In [8]: scoring = 'accuracy'
```

```
In [9]: from sklearn import model_selection
```

```
In [10]: kfold = model_selection.KFold(n_splits=10, random_state=seed)

cv_results = model_selection.cross_val_score(GaussianNB(), X_train, Y_train, cv=kfold, scoring=scoring)
cv_results_V = model_selection.cross_val_score(GaussianNB(), X_validation, Y_validation, cv=kfold, scoring=scoring)

print('The accuracy of the Naive Bayes classifier on training data is {:.2f}'.format(cv_results.mean()))
print('The accuracy of the Naive Bayes classifier on test data is {:.2f}'.format(cv_results_V.mean()))
```

```
The accuracy of the Naive Bayes classifier on training data is 0.97
The accuracy of the Naive Bayes classifier on test data is 0.88
```

05

XGBoost

Let's Get
Started

XGBoost

XGBoost — Extreme Gradient Boosting เป็น model ที่นำเอา Decision Tree มา train ต่อๆ กันหลายๆ tree โดยที่แต่ละ decision tree จะเรียนรู้จาก error ของ tree ก่อนหน้า ทำให้ความแม่นยำของการ ทำ prediction(ทำนาย) จะแม่นยำมากขึ้น เรื่อยๆ เมื่อมีการเรียนรู้ของ tree ต่อเนื่องกันจนมีความลึกมากพอ และ model จะหยุดเรียนรู้เมื่อไม่เหลือ pattern ของ error จาก tree ก่อนหน้าให้เรียนรู้แล้ว

XGBoost

```
In [1]: import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import sklearn.datasets as datasets  
%pylab inline
```

Populating the interactive namespace from numpy and matplotlib

```
In [2]: iris = datasets.load_iris()  
XX = pd.DataFrame(iris.data, columns=iris.feature_names)  
yy = pd.DataFrame(iris.target)
```

```
In [3]: from sklearn.model_selection import train_test_split  
  
X_train, X_test, y_train, y_test = train_test_split(XX, yy, test_size=0.3, random_state=0)  
  
print('There are {} samples in the training set and {} samples in the test set'.format(X_train.shape[0], X_test.shape[0]))
```

There are 105 samples in the training set and 45 samples in the test set

```
In [4]: from sklearn.preprocessing import StandardScaler  
from sklearn.model_selection import cross_val_score  
  
sc = StandardScaler()  
sc.fit(X_train)  
X_train_std = sc.transform(X_train)  
X_test_std = sc.transform(X_test)
```

Import

- Numpy
- pandas
- Sklearn datasets

Load data

ต่อไปเมื่อสร้างของเรียกชื่อจากคลาส StandardScaler จากนั้นก็มาอุบเร็กท์ตัวนี้ไปเพื่อคิดเบริ่ง จำกว่าเมื่อเรา fit เพื่อนำเข้ามูลน้ำหน้าค่าจะถูกเปลี่ยนและล้วนเป็นแบบมาตรฐานเพื่อนำมาใช้ในการแปลงต่อไป จนกว่าจะถูก transform เพื่อทำการแปลงค่าที่เป็นค่าที่ปรับเป็นมาตรฐานแล้ว ผลที่ได้คือได้ X_train_std และ X_test_std ที่คำนวณเองได้

XGBoost

```
In [5]: #Applying XGBoost
import xgboost as xgb

xgb_clf = xgb.XGBClassifier()
xgb_clf = xgb_clf.fit(X_train_std, y_train)

print('The accuracy of the XGBoost classifier on training data is {:.2f}'.format(xgb_clf.score(X_train_std, y_train)))
print('The accuracy of the XGBoost classifier on test data is {:.2f}'.format(xgb_clf.score(X_test_std, y_test)))

The accuracy of the XGBoost classifier on training data is 1.00
The accuracy of the XGBoost classifier on test data is 0.98
```

06

Random forest

Let's Get
Started

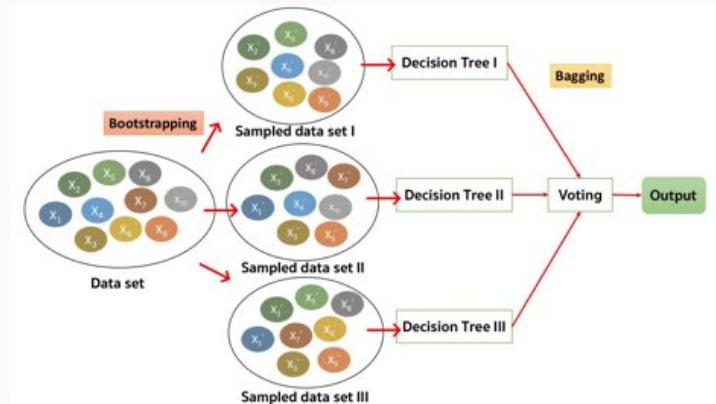
Random forest

Random forest สร้าง model จาก Decision Tree หลายตัว model ป้องกัน

(ตั้งแต่ 10 model ถึงมากกว่า 1000 model) โดยแต่ละ model จะได้รับ data set ไม่เหมือนกัน

ซึ่งเป็น subset ของ data set ทั้งหมด ตอนทำ prediction ก็ให้แต่ละ Decision Tree ทำ prediction

ของโครงข้อมูล และคำนวณผล prediction ด้วยการ vote output ที่ ถูกเลือกโดย Decision Tree มากที่สุด



Random forest

```
In [1]: import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import sklearn.datasets as datasets  
%pylab inline
```

Populating the interactive namespace from numpy and matplotlib

```
In [2]: iris = datasets.load_iris()  
XX = pd.DataFrame(iris.data, columns=iris.feature_names)  
yy = pd.DataFrame(iris.target)
```

```
In [3]: from sklearn.model_selection import train_test_split  
  
X_train, X_test, y_train, y_test = train_test_split(XX, yy, test_size=0.3, random_state=0)  
  
print('There are {} samples in the training set and {} samples in the test set'.format(X_train.shape[0], X_test.shape[0]))
```

There are 105 samples in the training set and 45 samples in the test set

```
In [4]: from sklearn.preprocessing import StandardScaler  
from sklearn.model_selection import cross_val_score  
  
sc = StandardScaler()  
sc.fit(X_train)  
X_train_std = sc.transform(X_train)  
X_test_std = sc.transform(X_test)
```

Import

- Numpy
- pandas
- Sklearn datasets

Load data

ต่อไปคือส่วนของการนำข้อมูลมาปรับเปลี่ยนจากคลาส **StandardScaler** จากนั้นก็นำข้อมูลเข้าไปให้โดยรีส์จากที่เมื่อเรา fit เพื่อนำข้อมูลมาหาค่าเฉลี่ยและส่วนเบี่ยงเบนมาตรฐานเพื่อนำมาใช้ในการแปลงต่อไป จากนั้นจึงให้มอง transform เพื่อทำการแปลงค่าให้เป็นค่าที่ปรับเน้นมาตรฐานแล้ว ผลที่ได้คือได้ **X_train_std** และ **X_test_std** ที่คำนวณเองได้

Random forest



```
In [5]: #Applying RandomForest
from sklearn.ensemble import RandomForestClassifier

#Create Random Forest object
random_forest = RandomForestClassifier()

#Train model
random_forest.fit(X_train_std, y_train)

#print performance
print('The accuracy of the Random Forest classifier on training data is {:.2f}'.format(random_forest.score(X_train_std, y_train)))
print('The accuracy of the Random Forest classifier on test data is {:.2f}'.format(random_forest.score(X_test_std, y_test)))

```

The accuracy of the Random Forest classifier on training data is 1.00
The accuracy of the Random Forest classifier on test data is 0.98

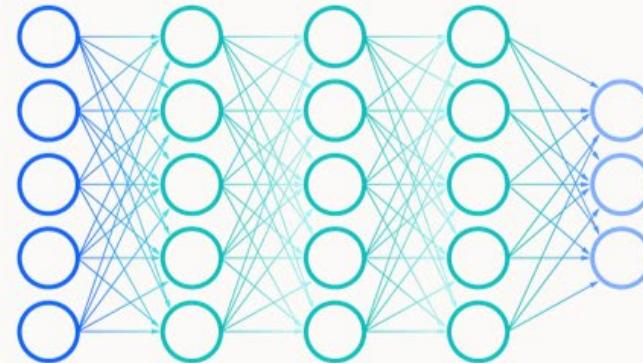


CNN Model

Neural
Network

Neural Network

Neural Network หรือ Artificial Neural Network คือ โครงข่ายประสาทเทียม เป็นสาขาหนึ่งของปัญญาประดิษฐ์ Artificial Intelligence (AI) เป็นแนวคิดที่ออกแบบระบบโครงข่ายคอมพิวเตอร์ ให้เลียนแบบการทำงานของสมองมนุษย์



Neural Network



In [1]: *#Neural Network Classification*

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

from sklearn.model_selection import train_test_split

from keras.models import Sequential
from keras.layers.core import Dense, Activation
from keras.utils import np_utils
```

Import

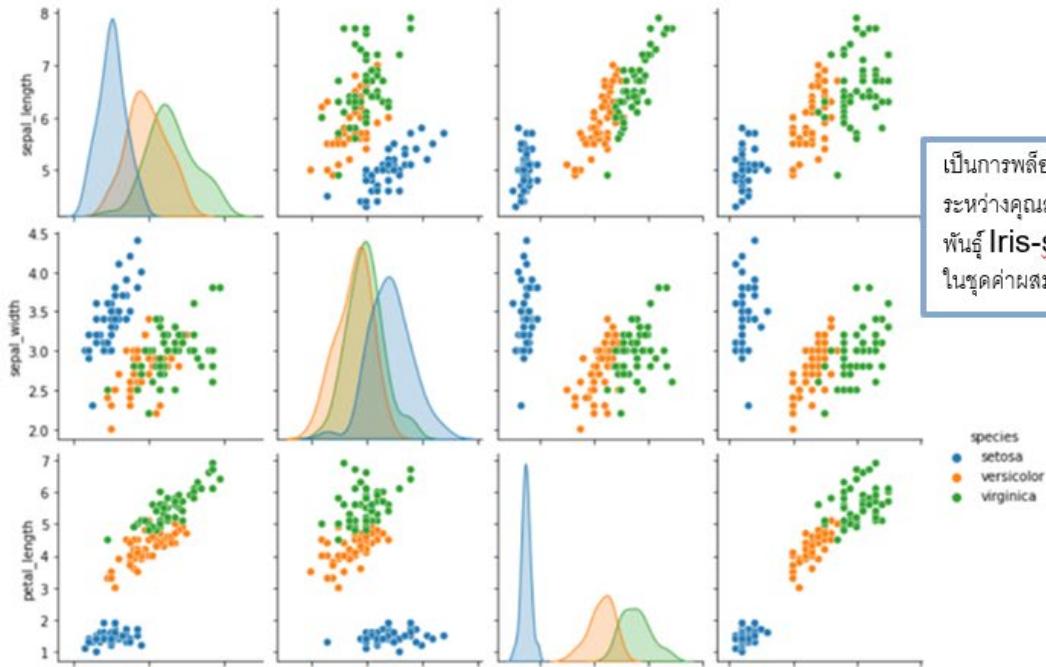
- Numpy
- Matplotlib
- seaborn
- Sklearn datasets
- keras

Neural Network



```
In [2]: iris = sns.load_dataset("iris")
sns.pairplot(iris, hue='species')

Out[2]: <seaborn.axisgrid.PairGrid at 0x21b74ec6850>
```

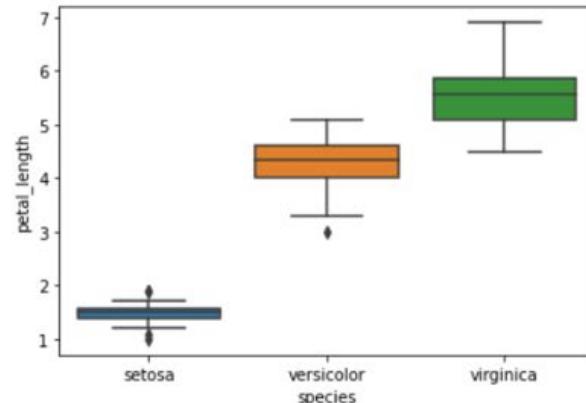


เป็นการพื้อตัวให้ดู ซึ่งแสดงความสัมพันธ์ของตัวแปร
ระหว่างคุณสมบัติแต่ละคู่ จากพื้อตัวคุณรู้จะเห็นว่าสาย
พันธุ์ Iris-setosa ถูกแยกออกจากอีกสองสายพันธุ์
ในชุดค่าผ่านทั้งหมด

Neural Network



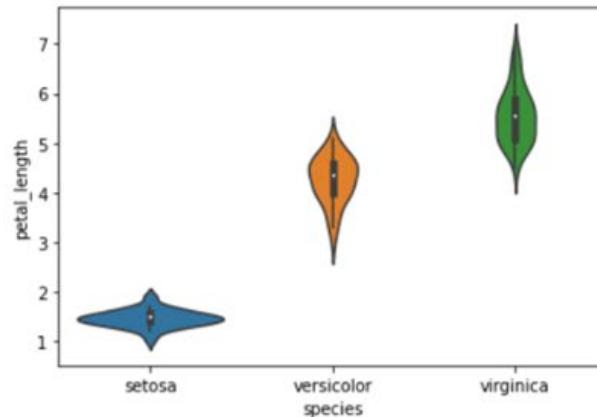
```
In [3]: sns.boxplot(x="species",y="petal_length",data=iris)  
plt.show()
```



Neural Network



```
In [4]: sns.violinplot(x="species",y="petal_length",data=iris)  
plt.show()
```



Neural Network



```
In [5]: X=iris.values[:, :4].astype(np.float)
y=iris.values[:, 4]
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,train_size=0.5,random_state=1)
```

```
In [6]: def one_hot_encode_object_array(arr):
    '''One hot encode a numpy array of objects'''
    uniques, ids = np.unique(arr, return_inverse=True)
    return np_utils.to_categorical(ids, len(uniques))

y_train_ohe=one_hot_encode_object_array(y_train)
y_test_ohe=one_hot_encode_object_array(y_test)
```

```
In [7]: model=Sequential()

model.add(Dense(16,input_shape=(4,)))
model.add(Activation("sigmoid"))

model.add(Dense(3))
model.add(Activation("softmax"))

model.compile(optimizer="adam",loss="categorical_crossentropy",metrics=["accuracy"])
```

Neural Network



```
In [8]: model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
=====		
dense (Dense)	(None, 16)	80
activation (Activation)	(None, 16)	0
dense_1 (Dense)	(None, 3)	51
activation_1 (Activation)	(None, 3)	0
=====		
Total params: 131		
Trainable params: 131		
Non-trainable params: 0		



Neural Network



```
In [9]: model.fit(X_train,y_train_ohe,epochs=30,batch_size=1,verbose=1)

loss, accuracy = model.evaluate(X_test, y_test_ohe, verbose=1)
print("The accuracy of the Neural Network classifier on test data is = {:.2f}".format(accuracy))

Epoch 1/30
75/75 [=====] - 0s 535us/step - loss: 1.2339 - accuracy: 0.3517
Epoch 2/30
75/75 [=====] - 0s 562us/step - loss: 1.1285 - accuracy: 0.4054
Epoch 3/30
75/75 [=====] - 0s 527us/step - loss: 1.0367 - accuracy: 0.4228
Epoch 4/30
75/75 [=====] - 0s 538us/step - loss: 1.0433 - accuracy: 0.6154
Epoch 5/30
75/75 [=====] - 0s 510us/step - loss: 1.0149 - accuracy: 0.6949
Epoch 6/30

Epoch 29/30
75/75 [=====] - 0s 586us/step - loss: 0.4249 - accuracy: 0.9168
Epoch 30/30
75/75 [=====] - 0s 519us/step - loss: 0.4205 - accuracy: 0.9525
3/3 [=====] - 0s 1ms/step - loss: 0.4017 - accuracy: 0.9867
The accuracy of the Neural Network classifier on test data is = 0.99
```





Thanks!

Do you have any questions?